Lecture 4:
Software Lifecycles

The Software Process

- Waterfall model
- Rapid Prototyping Cycle
- Phased Models:
  - Incremental Development
  - Evolutionary Development
  - Spiral Model
- V-model and Systems Engineering
- The ‘essential’ software process

Verification and Validation

systems development -common steps

planning

analysis

design

development

implementation

maintenance

systems development methodologies

wild west approach

waterfall model

prototyping (incremental, throwaway)

incremental development

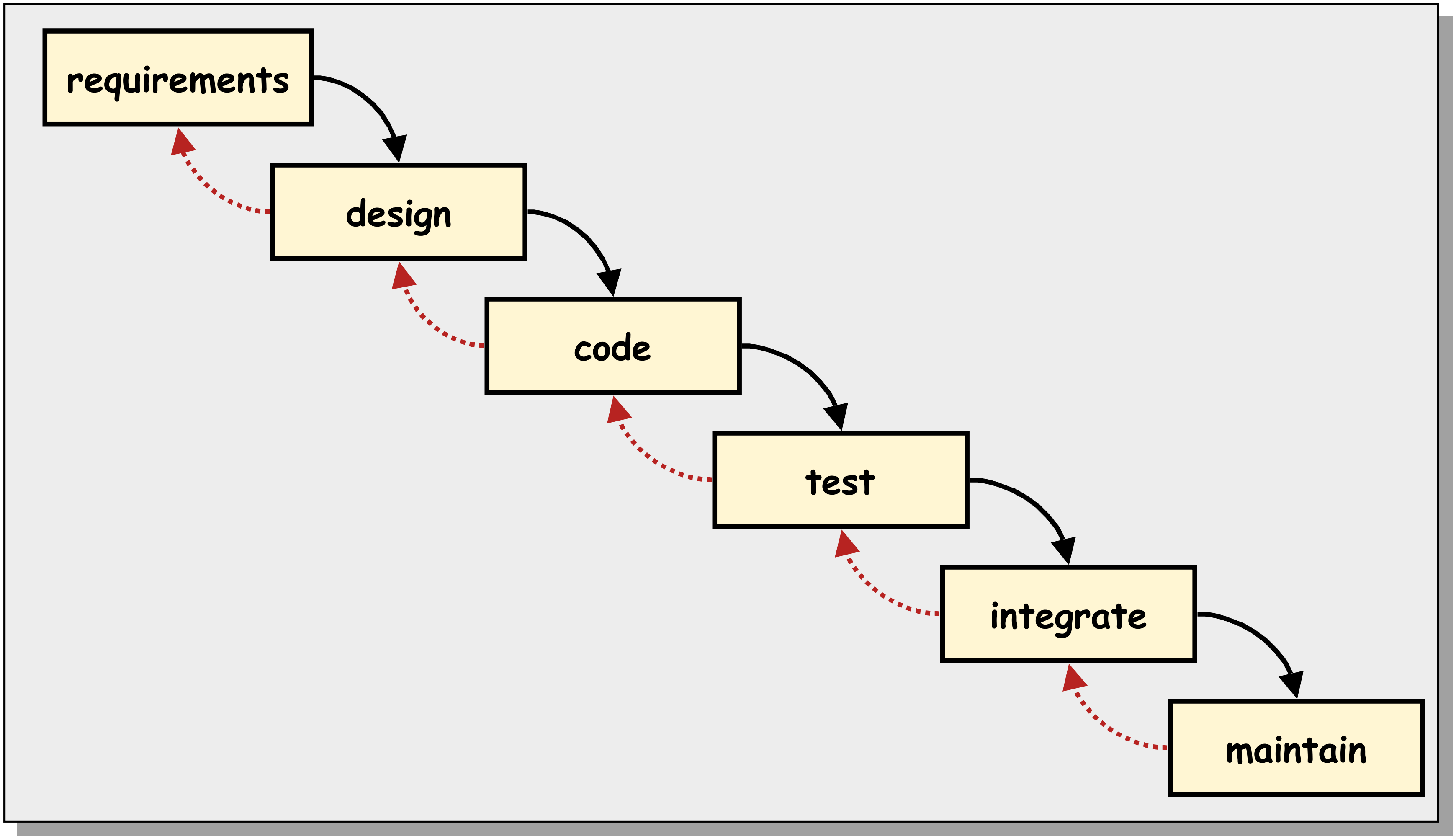
rapid application development

spiral model

agile/lightweight methods

Waterfall Model

Source: *Adapted from Dorfman, 1997, p7*
see also: *van Vliet 1999, p50*



Why not a waterfall?

Waterfall model describes a process of stepwise refinement

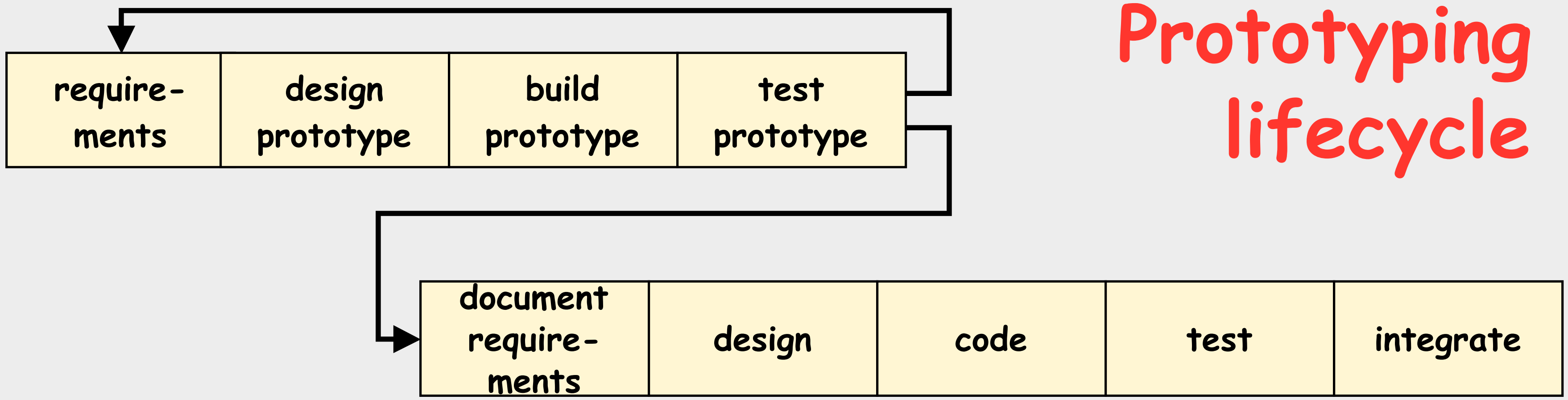
- Based on hardware engineering models
- Widely used in defense and aerospace industries

But software is different:

- No fabrication step
  - Program code is just another design level
  - Hence, no ‘commit’ step - software can always be changed…!
- No body of experience for design analysis (yet)
  - Most analysis (testing) is done on program code
  - Hence, problems not detected until late in the process
- Waterfall model takes a static view of requirements
  - ignores changing needs
  - Lack of user involvement once specification is written
- Unrealistic separation of specification from design
- Doesn’t accommodate prototyping, reuse, etc.

Source: *Adapted from Blum 1992, pp28-31*
see also: *van Vliet 1999, p50-1*

Prototyping lifecycle



Prototyping is used for:

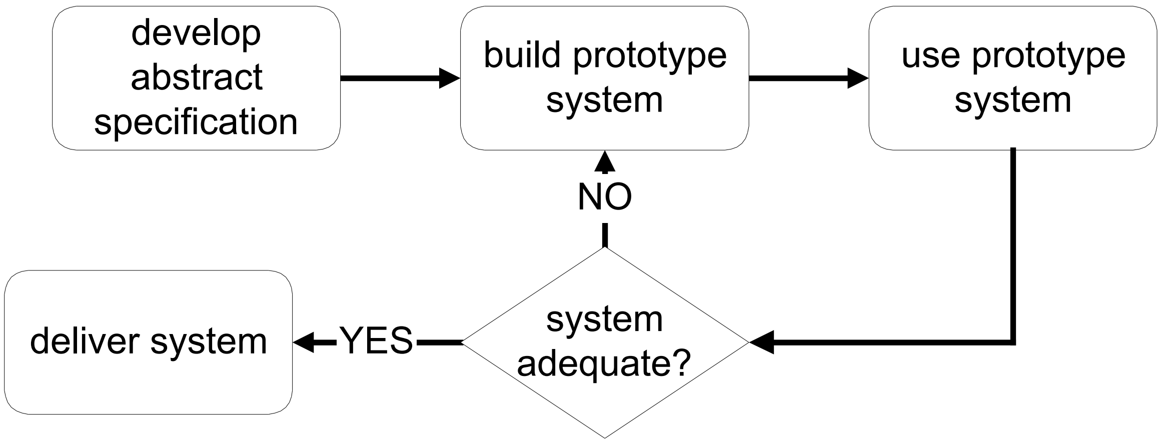
- understanding the requirements for the user interface
- examining feasibility of a proposed design approach
- exploring system performance issues

Problems:

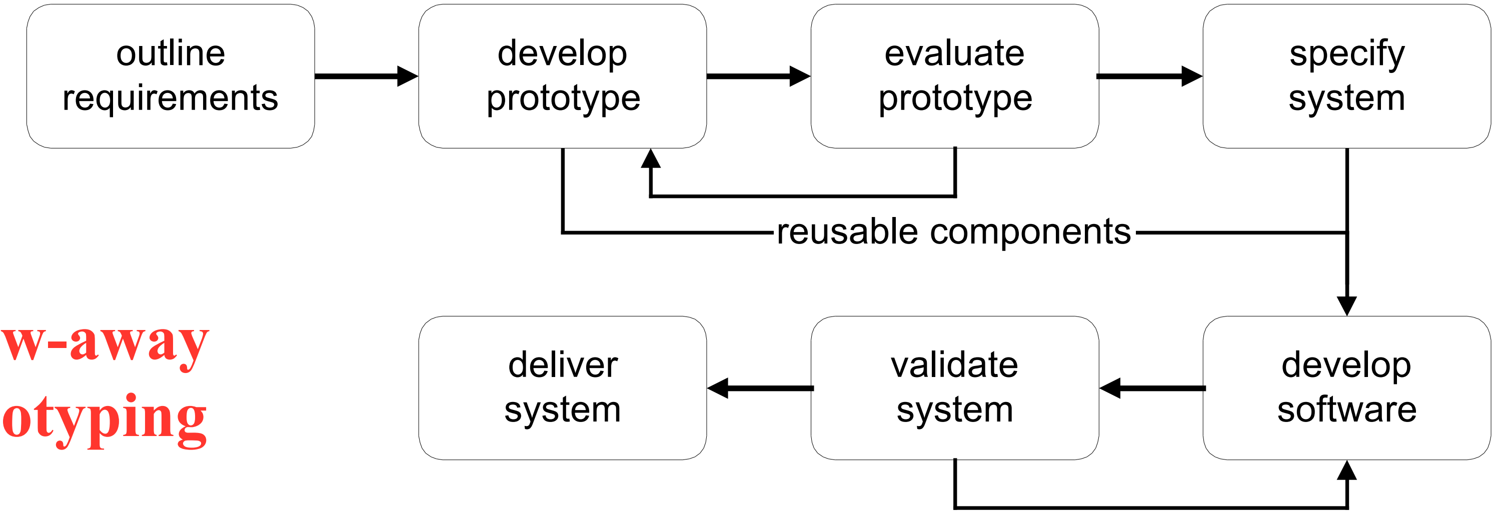
- users treat the prototype as the solution
- a prototype is only a partial specification

Source: *Adapted from van Vliet 1999, p53*

evolutionary prototyping



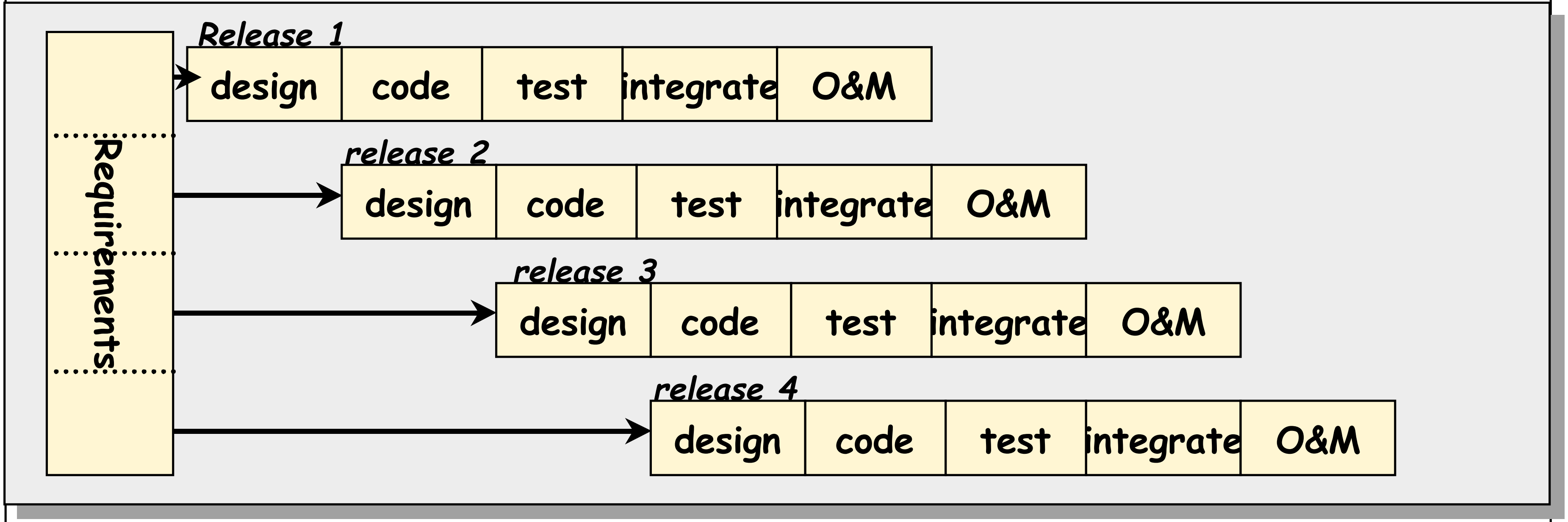
throw-away prototyping



Sommerville fig 8.3 and 8.5

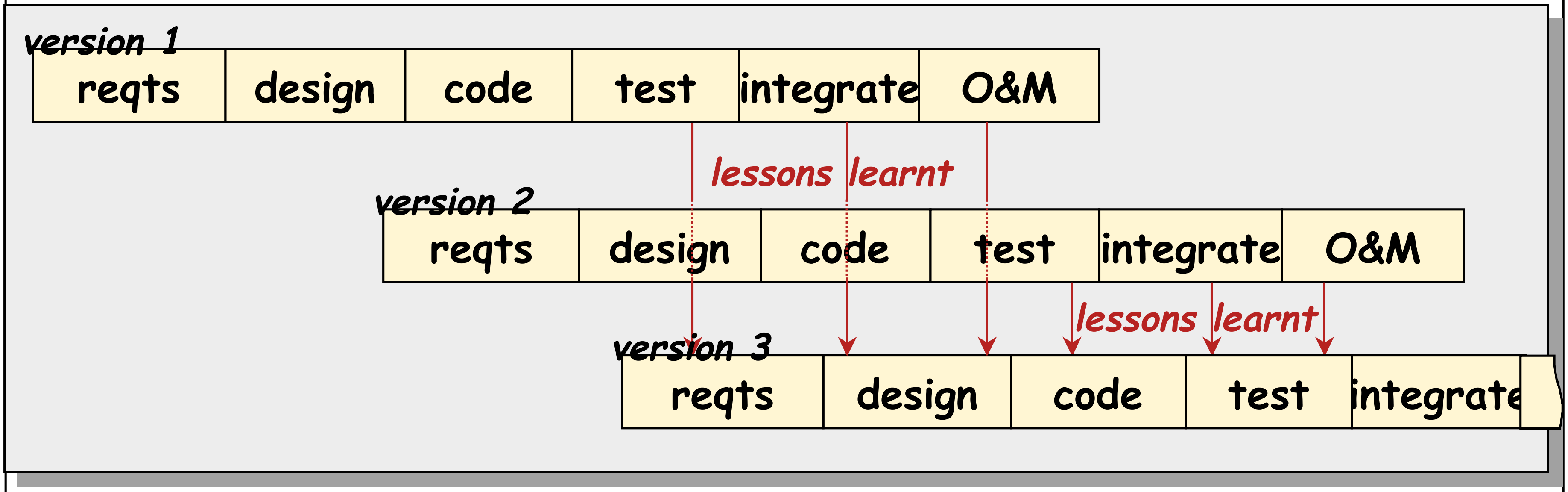
Source: *Adapted from Dorfman, 1997, p10*
see also: *van Vliet 1999, p56*

Phased Lifecycle Models



Incremental development

(each release adds more functionality)

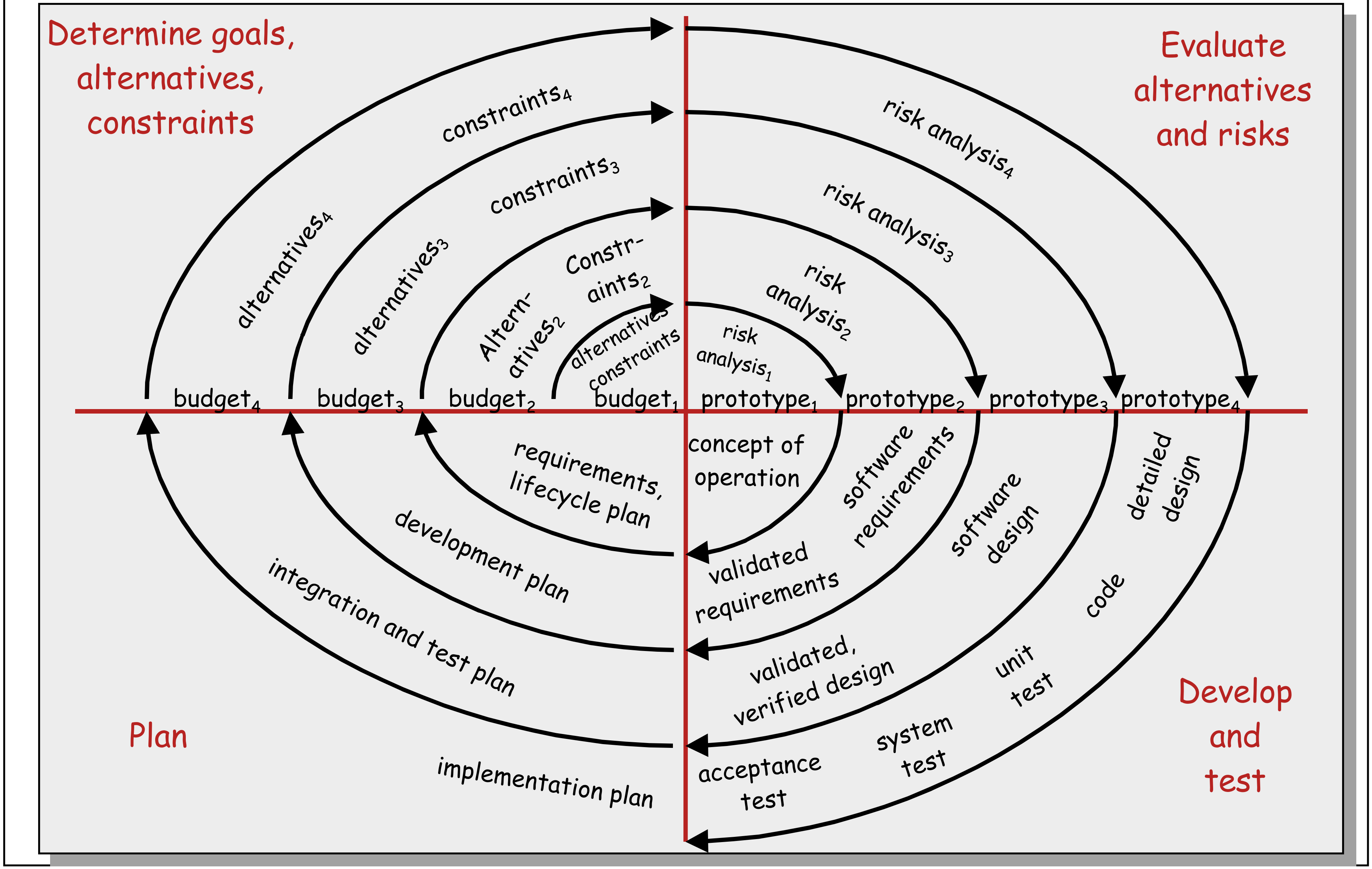


Evolutionary development

(each version incorporates new requirements)

The Spiral Model

Source: *Adapted from Pfleeger, 1998, p57*
see also: *van Vliet 1999, p63*



Comments on phased models

Incremental development

avoids 'big bang' implementation

but:

- assumes all requirements known up-front

Evolutionary development

allows for lessons from each version to be incorporated into the next

but…

- hard to plan for versions beyond the first;
- lessons may be learnt too late

Spiral model

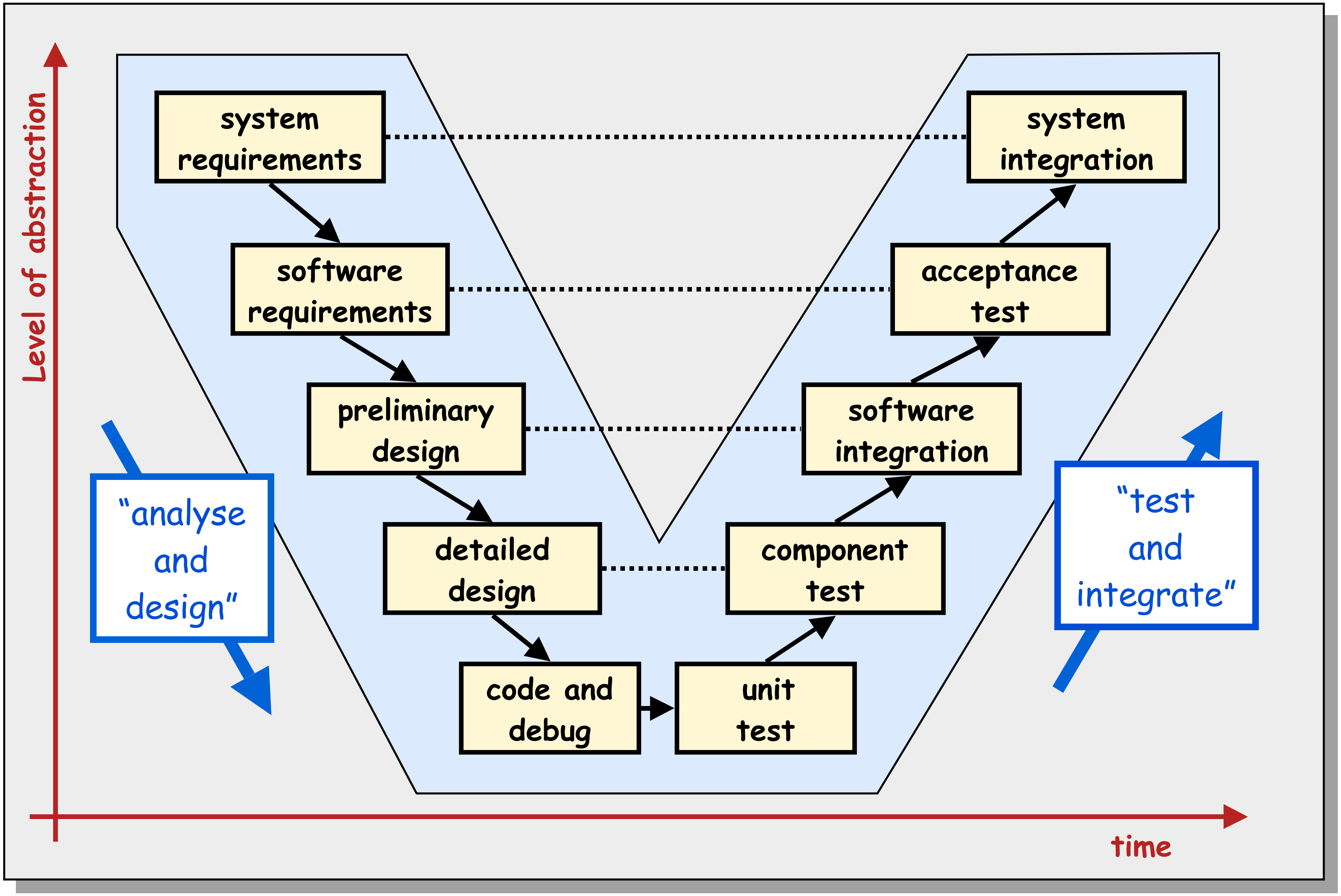
incorporates prototyping and risk analysis

but…

- cannot cope with unforeseen changes (e.g. new business objectives)
- not clear how to analyze risk

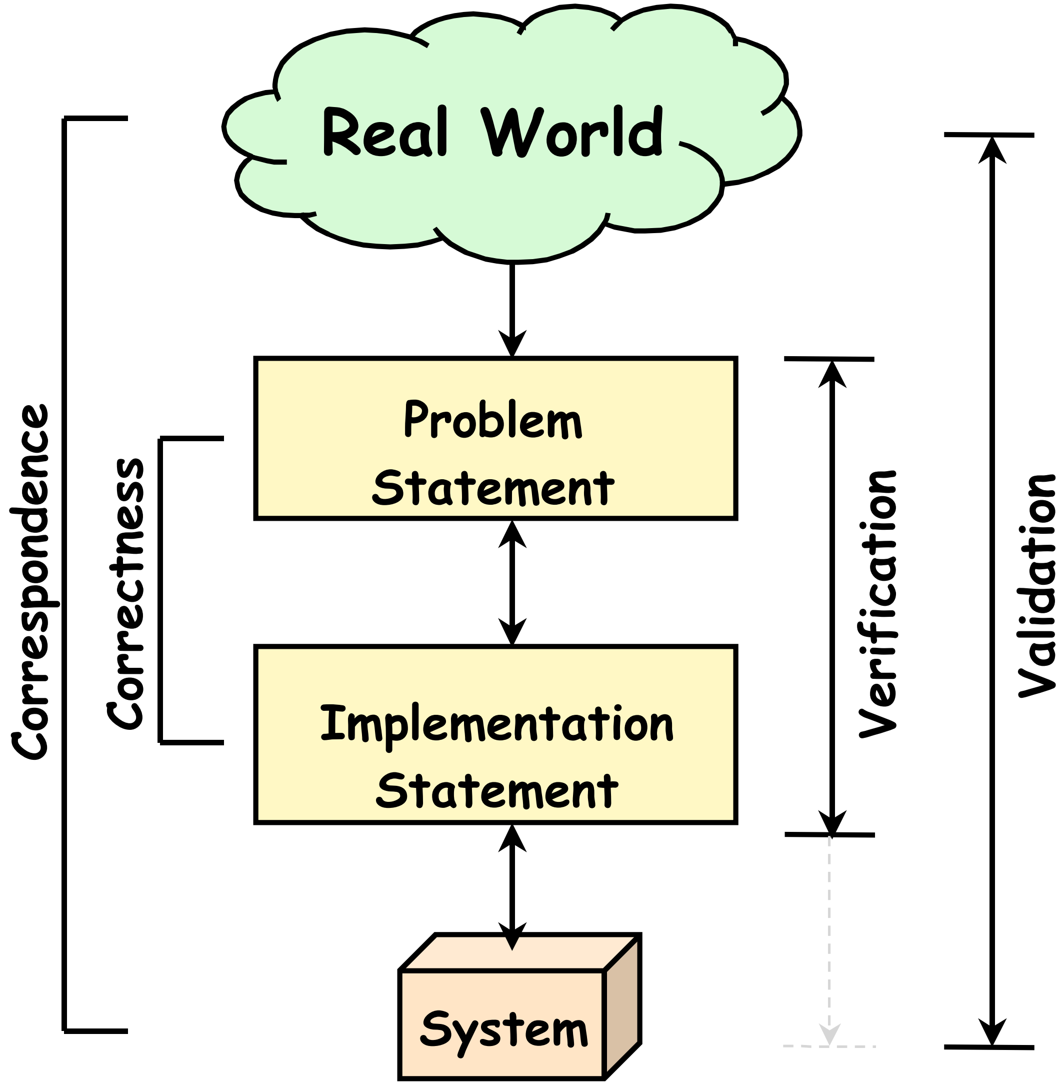
V-Model

Source: *Adapted from Forsberg & Mooz 1997*

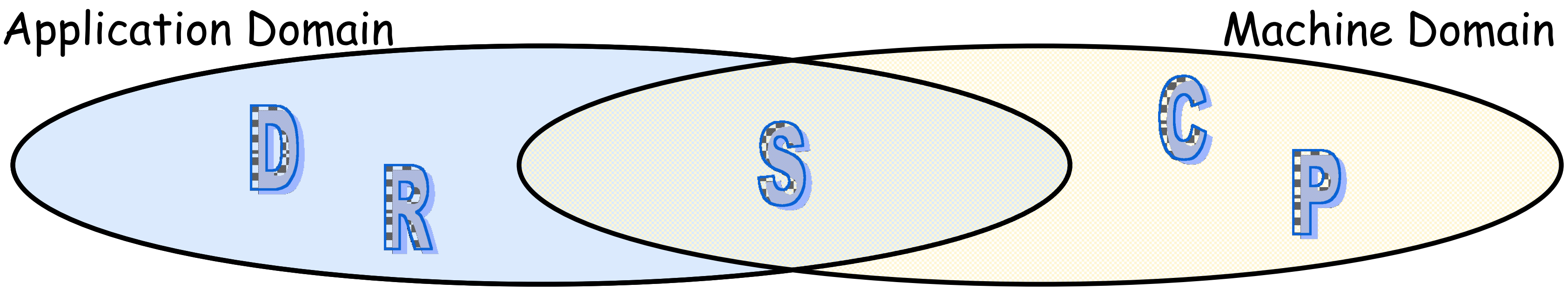


The “essential” software process

Source: *Adapted from Blum, 1992, p32*
see also: *van Vliet p11*



Verification and Validation



For V&V, we need to worry about:

- The properties of the computer hardware (C)
- The properties of the program (P)
- The properties of the machine in the application domain (the specification, S)
- The properties of the domain, independent of the machine (D)
- The requirements for the machine (R)

Demonstrating that P satisfies R is then a two step process:

- Do C and P imply S? *(Verification)*
- Do S and D imply R? *(Validation)*

Source: *Adapted from Jackson, 1995, p170-171*

Validation Example

Source: *Adapted from Jackson, 1995, p172*

Requirement R:

"Reverse thrust shall only be enabled when the aircraft is moving on the runway"

Domain Properties D:

Wheel pulses on if and only if wheels turning

Wheels turning if and only if moving on runway

Specification S:

Reverse thrust enabled if and only if wheel pulses on

S + D imply R

But what if the domain model is wrong?

Summary

Software is different

- many assumptions from other engineering models don't apply
- there is no fabrication step
- the underlying science of software behaviour is not well developed
- (software engineering is still an immature discipline)

Many different views of the software process

- waterfall model is too rigid (doesn't allow for change)
- other models incorporate prototyping, evolution, risk, etc.
- no lifecycle model is perfect

Essential process:

- describe the problem
- describe the solution
- verify (does the solution solve the stated problem?)
- validate (did we solve the right problem?)

References

van Vliet, H. “Software Engineering: Principles and Practice (2nd Edition)” Wiley, 1999.

Chapter 3 provides a very good overview of lifecycle models.

Blum, B. “Software Engineering: A Holistic View”. Oxford University Press, 1992.

Dorfman, M. “Requirements Engineering”. In Thayer, R. H and Dorfman, M. (eds.) “Software Requirements Engineering, Second Edition”. IEEE Computer Society Press, 1997, p7-22

Forsberg, K and Mooz, H. “System Engineering Overview”. In Thayer, R. H and Dorfman, M. (eds.) “Software Requirements Engineering, Second Edition”. IEEE Computer Society Press, 1997, p44-72

Jackson, M. “Software Requirements & Specifications: A Lexicon of Practice, Principles and Prejudices”. Addison-Wesley, 1995.

Pfleeger, S. “Software Engineering: Theory and Practice”. Prentice Hall, 1997.